НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050103 “Програмна інженерія’’

на тему: Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Виконав: студент 4 курсу, групи ТВ-51

Шикер Богдан Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. в. Матях Сергій Володимирович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050103 “Програмна інженерія’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Шикер Богдану Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)”
керівник роботи ст. в. Матях Сергій Володимирович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 22 03 2019р. № 1325-с

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи ASP.Net Core проект
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби аналізу сонячної активності, спроектувати архітектуру системи аналізу сонячної активності, розробити програмне забезпечення, розробити інтерфейс користувача
5. Перелік ілюстраційного матеріалу
1. Актуальність 2. Мета та завдання роботи 3. Функції доадтку 4. Опис функціональності системи5. Архітектура програмного комплексу 6. Опис архітектури серверу 7. Використані програмні засоби 8. Інтерфейс модулю керування картою 8.Меню керування показниками сонячної радіації 9. Меню

керування видами сонячних панелей 10. Висновки
Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Шикер Б.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матях С. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи була розробка серверної частини та адміністративної панелі для інтерактивної карти альтернативних джерел енергії України. Система дозволяє дослідити метеорологічні дані по території України. Завдяки цьому користувачі можуть провести аналіз ефективності використання сонячної панелей та геліосистем в будь-якій частині України. Адміністративна панель, дозволяє додавати маркери з інформацією на карту, редагувати значення сонячної активності в регіонах та розраховувати прибуток від використання сонячних панелей та геліосистем.

Записка містить 76 сторінки, 19 картинок та 8 посилань

ABSTRACT

The purpose of the work was to develop the back-end part and admin panel for the interactive map of alternative energy sources of Ukraine. The system allows to explore meteorological data on Ukraine territory. Due to this, users can analyze the efficiency of solar panels and heliosystems at any part of Ukraine. Admin panel allows to add markers with information to the map, edit values of solar activity in regions and calculate profit of solar station or heliosystem.

The note contains 76 pages, 19 images and 8 references

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних;

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

ВСТУП

Одним з найбільш перспективних напрямів розвитку світової енергетики в даний час є використання відновлюваних джерел енергії, що знімає ряд проблем, пов'язаних з використанням традиційних палив. Кінець другого тисячоліття характеризується інтенсивним ростом обсягів використання енергії відновлюваних джерел енергії у світі. Зважаючи на проблеми глобального потепління, все більше людей замислюється над можливістю переходу на екологічно чисті, відновлюванні джерела енергії. Дана система націлена допомогти зацікавленим у цьому людям.

Для реалізації систем сонячного енергопостачання необхідно врахування комплексу додаткових факторів для вибору раціональних промислових площадок розміщення фотоелектричних та геліоенергетичних станцій, а саме: економічних, екологічних, та соціальних особливостей кожного з регіонів України.

Базуючись на проведених дослідженнях, зроблено висновок, що системи, яка б в повному обсязі дозволяла дослідити та проаналізувати сонячну активність та ефективність її використання як джерела енергії-немає. А також визначено, що використання інтерактивної онлайн-карти сприяє кращому вивченні галузі, тому дана є актуальною.

Тому запропоновано розробити web-додаток, який використовуючи дані з метеостанцій (які будуть впливати на коефіцієнт корисної дії електростанцій), дозволить проводити аналіз ефективності використання сонячних панелей та геліосистем в регіонах України.

Користувачем системи буде кожен відвідувач сайту, який зацікавлений в альтернативних джерелах енергії, але не знає деталі клімату в своєму регіоні. Він отримує можливість на карті України вибирати будь-яку точку та завдяки метеоданим цієї області розрахувати, яке обладнання потрібне для отримання необхідної потужності.

Записка містить 7 розділів.

У першому розділі описується постановка задачі відображення сонячної активності на території України.

У другому розділі описується характеристика сонячної радіації.

У третьому розділі описані підходи використання сонячної енергії.

У четвертому розділі описані способи аналізу сонячної активності.

У п'ятому розділі вказуються основні засоби розробки даної системи.

У шостому розділі дано опис реалізованого програмного продукту і його архітектури.

У сьомому розділі описано роботу користувача з системою.

1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ ІНТЕРАКТИВНОЇ КАРТИ АЛЬТЕРНАТИВНИХ ДЖЕРЕЛ УКРАЇНИ.

З темпами розвитку сучасного світу та технології, важливо ефективно систематизувати та аналізувати дані. Масиви даних є досить великими, тому для швидкого виділення з них корисної інформації зручно використовувати інструменти візуалізації. Прикладом таких даних э звіти про сонячну активність в різних куточках планети.

Поняття сонячної активності в сучасній науці пов'язане з терміном «сонячна інсоляція». Під інсоляцією розуміється кількість радіації, отримана протягом одного світлового дня, або, просто кажучи, ступінь «опромінення» 1 м.кв. землі за конкретний проміжок часу. В даному контексті не варто лякатися терміну «радіація», оскільки тут сонячне опромінення є потенційно корисним енергетичним ресурсом, а не джерелом небезпеки.

Зібрати точні дані для аналізу є нелегкою задачею. Складність полягає в тому, що точно виміряти кількість потрапив на землю світла неможливо, адже процес радіаційного опромінення піддається впливу безлічі факторів, наприклад:

- висота ділянки над рівнем моря і, відповідно, віддаленість сонця від даної місцевості;
- висота ділянки над рівнем моря і, відповідно, віддаленість сонця від даної місцевості;
- пору року (також вносить корективи в величину відстані сонця від землі);
- погодні умови (хмарність, тумани тощо.);
- кут падіння сонячних променів (різниться за часом доби).

Зібравши данні, необхідно представити їх у вигляді моделі, зрозумілої кожному користувачу. Тому було запропоновано дослідити методи моделювання сонячної радіації для вирішення поставленої задачі.

Як результат дослідження, потрібно створити веб-додаток, основними функціями якого є:

- відображення інтуїтивно зрозумілої карти сонячної радіації по Україні;
- калькулятор розрахунку видобутку електроенергії з використанням різних видів сонячних панелей в кожному місяці;
- калькулятор розрахунку видобутку об’єму гарячої води з використанням різних видів геліосистем в кожному місяці;
- створення на карті міток з інформацією про сонячні станції та геліосистеми;
- внесення даних про сонячну активність в різних регіонах та можливість їх редагування.

2. ХАРАКТЕРИСТИКА СОНЯЧНОЇ ЕНЕРГІЇ

Сонячна радіація – це енергія, що випромінюється Сонцем. Вона- основне джерело енергії для всіх атмосферних процесів. Сонячна радіація визначається кількістю тепла, що вона виділяє та вимірюється в мегаджоулях на 1 кв. м. Сонячні промені здатні досягати земної поверхні та проінкати через всі шари атмосфери, які частково поглинають, розсіюють і відбивають сонячну радіацію. Тепловий потік сонячного випромінювання, що досягає Землі є надзвичайно великим. Він перевищує сумарне використання всіх видів паливно-енергетичних ресурсів у світі в більш ніж 5000 разів.

Головними перевагами використання сонячної енергії є її довговічність та повна екологічна чистота. Сонячна енергія доступна в кожній точці планети, окрім полюсів. Отже, на всій земній кулі лише ніч та хмари можуть завадити користуватися нею постійно. Завдяки цьому, цей вид енергії неможна монополізувати, на відміну від нафти та газу. На жаль, вартість 1 кВт • год. сонячної енергії перевищує вартість енергії отриманої традиційними методами. Тільки одна п’ята частина сонячного світла конвертується в електричний струм, але ця частина продовжує зростати завдяки роботі учених та інженерів всього світу.

Так як сонячне випромінювання розподілене по величезній площі (тобто, має низьку густину), кожна установка для прямого використання сонячної енергії мусить мати збираючий пристрій з достатньою площею поверхні. Найпростіший пристрій такого роду — плоский колектор. Це чорна плита, яка добре ізольована знизу. Вона покривається склом чи пластмасою, яка пропускає світлові промені, але не пропускає інфрачервоне теплове випромінювання. Між плитою та склом розміщують чорні трубки, в яких тече масло, вода, повітря чи сірчистий ангідрид. Сонячне проміння, проникаючи в колектор, поглинається чорними трубками і плитою та нагріває робочу речовину в трубках.

2.1 Види сонячних променів

Сонячні промені, які досягають поверхні Землі, підрозділяють на два види: прямі і розсіяні.

Прямі сонячні промені - це, як видно з назви, промені, які безпосередньо з поверхні Сонця досягають поверхні Землі. Потужність прямого сонячного випромінювання залежить від чистоти (ясності) атмосфери, висоти Сонця над обрієм (залежить від географічної широти і часу дня), а також від положення поверхні по відношенню до Сонця.

Розсіяні сонячні промені надходять з верхніх шарів атмосфери і залежать від того, яким чином прямі сонячні промені відбиваються від Землі і навколишнього середовища. Завдяки повторюваного процесу відображення між покритою снігом поверхнею Землі і нижньою стороною хмар потужність розсіяного сонячного випромінювання може досягати великих значень. Сонячні промені несуть з собою невичерпний потік енергії. Вони постійно доставляють на Землю більшу кількість енергії, ніж нам сьогодні необхідно. Щільність сонячних променів в космосі дорівнює приблизно 1,4 кВт / м2. З них близько 30% відбивається назад в космос, так і не досягнувши Землі. На поверхні Землі щільність сонячних променів складає 1 кВт / м2. Сонячна енергія, що досягла поверхні Землі, несе з собою тепло, випаровує воду, утворює вітер і рух води в морях, дає життя рослинам. Та сонячна енергія, яка безпосередньо не поглинається на Землі, відбивається в космос. Земля знаходиться в постійному тепловому балансі з навколишнім середовищем. Якби цього не відбувалося, то Земля нагрівалася б все сильніше і в результаті всяке життя на ній виявилася б неможливою.

2.2 Проблема збереження сонячної енергії

Запаси енергії великі, якщо не сказати, необмежені. Проблема полягає в тому, що ми отримуємо найбільшу кількість сонячної енергії влітку, тобто в той час, коли найменше в ній потребуємо. А взимку, коли нам потрібна велика кількість енергії, Сонце світить тільки короткий період вдень і під низьким кутом.

Отже, ситуація така: влітку - велика кількість енергії та мінімальні потреби, а взимку - незначна кількість енергії, але, навпаки, величезні потреби. Відповідь до вирішення цієї проблеми проста: треба накопичувати енергію влітку і використовувати її взимку. Як це зробити? Запропоновано два види накопичення. Перше, накопичення тепла за допомогою, наприклад, сонячних панелей. В цьому випадку ми хочемо зберегти тепло протягом декількох днів, можливо, тижнів. По-друге, накопичення енергії, коли нам необхідно зберегти запаси енергії на довгий термін. Це загальнонаціональна проблема, яку на даний момент намагаються вирішити за допомогою розробки потужних та надійних акумуляторів.

2.3 Вимірювання та моделювання сонячного випромінювання

Наземні датчики, встановлені на професійних метеорологічних станціях і супутникові метеорологічні моделі, є двома основними підходами для отримання даних про сонячні ресурси. Ці два методи доповнюють один одного. Високоточні наземні сонячні датчики забезпечують вимірювання на високій частоті з високою точністю (якщо вони професійно встановлені та підтримуються). Як правило, висока точність і надійні дані сонячних датчиків доступні лише для обмеженого числа конкретних місць, і вони представляють лише конкретні та обмежені періоди часу. Супутникові сонячні моделі здатні постійно контролювати великі географічні райони в режимі реального часу.

Перевагою супутникових моделей є те, що вони представляють історію від 10 до більш ніж 20 років (залежно від регіону). Обмежуючою особливістю даних, отриманих від супутникових джерел, є їх часовий і просторовий дозвіл і менша точність (у порівнянні з високоякісними наземними вимірами). Вимірювання, отримане наземними датчиками, сприяє зменшенню невизначеності модельованих значень, тоді як моделі сонячного випромінювання взаємно забезпечують економічний спосіб контролю якості вимірювань, що проводяться наземними метеорологічними станціями.

Під час планування та експлуатації сонячних енергетичних проектів, рекомендований підхід полягає у використанні супутникових моделей для покриття потреб сонячних ресурсів на всіх етапах життєвого циклу проекту і в той же час встановлення наземних сонячних та метеорологічних датчиків (метеостанцій) на існуючих або запланованих масштабних проектів та інших стратегічних місцях. Якість даних сонячних ресурсів є критично важливою для економічної та технічної оцінки сонячних електростанцій. Розуміння невизначеності та управління ризиком, пов’язаним з погодою, є важливим для успішного планування та експлуатації сонячних енергогенеруючих активів.

3. СПОСОБИ ВИКОРИСТАННЯ СОНЯЧНОЇ ЕНЕРГІЇ

Сонячна енергія може бути перетворена в корисну енергію за допомогою використання активних або пасивних сонячних енергетичних систем.

3.1 Пасивні системи використання сонячної енергії

Найбільш ефективне використання пасивної сонячної енергії – це за допомогою ємності темного кольору та води. Темний колір, накопичуючи сонячну енергію, перетворює її в тепло та нагріває воду.

Проте, є більш прогресивне та методологічне використання сонячної енергії. Розробляються будівельні технології, які забезпечують реалізацію будинків, обробку кліматичних умов, підпорядкування будівельних матеріалів, щоб максимально використовувати сонячну енергію для опалення або охолодження, освітлення забудовника. За таким проектом будівля є колектором, який накопичує сонячну енергію.

Таким чином, було побудовано будинок на півночі Італії. В одній кімнаті вікна зроблені з слюди. Виявилося, що в цій кімнаті тепло і на неї обіцянку повинно бути менше. В цьому випадку слюда була як утеплювач. Сучасні будівельні конструкції впливають на географічне положення забудовника. Таким чином, велика кількість вікон, які виходять на північну сторону, дозволяє північних регіонах перевищувати рівень сонячного світла і тепла і обмежувати кількість вікон східної і західної сторонами. У таких будівлях орієнтація вікон і розташування, теплове навантаження і теплоізоляція - основна система проектування в проекті.

Такі будівлі екологічно чисті, енергонезалежні та комфортні. У приміщеннях багато природного світла, зв'язок з природою сприймається більш повно, а електрику економічно вигідно. Тепло в таких будівлях зберігається за рахунок обраних теплоізоляційних матеріалів, плакованих під землею. Ці перші

«сонячні» будівлі набули популярності в Америці після Другої світової війни. Згодом через більш низьких цін на нафту інтерес до проектування таких будинків дещо знизився. Однак через глобальної екологічної кризи це важливо для екологічних проектів з системами відновлюваної енергії.

3.2 Активні системи використання сонячної енергії

В основі активних систем використання сонячної енергії лежать сонячні колектори. Колектор, поглинаючи сонячну енергію, перетворює її в тепло, яке через теплоносій обігріває будівлі, нагріває воду, може перетворити його в електричну енергію і т.п. Сонячні колектори можуть застосовуватися у всіх процесах в промисловості, сільському господарстві, побутових потребах, де використовується тепло.

3.3 Види колекторів

Повітряний сонячний колектор є найпростішою формою сонячних колекторів (рис. 3.3.1). Його конструкція надзвичайно проста і нагадує ефект звичайного потоку, який розташований на будь-якому віддаленому ділянці. Провести невеликий експеримент. У зимовий сонячний день, розмістимо будь-який предмет на підвіконні так, щоб він потрапив на сонячні промені і через деякий час поклав долоню на нього. Ви відчуєте, що цей предмет став теплим. А за вікном може бути - 20! Ось принцип і заснована робота сонячного повітряного колектора.

Основним елементом колектора є теплоізоляційна плита, виготовлена з будь-якого матеріалу, який добре працює при нагріванні. Пластина забарвлена в темний колір. Сонячні промені проходять через прозору поверхню, нагріваючи пластину, а потім повітря пропускає тепло в приміщення. Повітря проходить через природну конвенцію або з вентилятором, що покращує теплообмін.

Рисунок 3.1 – Повітряний сонячний колектор

Але такі системи недоступні, оскільки вимагають додаткових витрат на роботу вентиляторів. Ці колектори працюють протягом дня, тому вони не можуть замінити основні джерела енергії повністю. Однак, якщо ви можете під’єднати колектор в основне джерело нагрівання або вентиляції, його ефективність незмірно зростає. Сонячні повітряні колектори також можуть бути використані для забезпечення гарячої води, що знижує її ціну до 40 євро на кубічний метр.

Сонячні колектори можуть бути плоскими (рис. 3.3.2) і вакуумними (рис. 3.3.3). Колектор складається з елемента, що поєднується з енергією, покриттям (зі зменшеним вмістом металу), трубопроводом і термореактивним шаром. Прозоре покриття захищає корпус від несприятливих кліматичних умов. Всередині корпусу поглинач панелі сонячної енергії (абсорбера) з'єднаний з теплоносієм, який циркулює по трубах. Трубопровід може бути або сітковим, або серпантиновим. Теплоносій рухається по них від входу до вихідних сопел,

поступово нагріваючись. Поглинаюча панель виконана з металу, що добре проводить тепло (алюміній, мідь).

Колектор захоплює тепло, перетворюючи його в теплову енергію. Такі колектори можуть бути вставлені в дах або розміщені на даху будівлі, але їх можна розмістити окремо.

Рисунок 3.2- Плоский сонячний колектор

Вакуумні колектори можна використовувати цілий рік. Основним елементом колекторів є вакуумні трубки. Кожна з них складається з двох скляних трубок. Труби виконані з боросилікатного скла, а інтер'єр покритий спеціальним покриттям, що забезпечує поглинання тепла з мінімальним відображенням. З простору між трубами відкачується повітря. Газ барію використовується для підтримки вакууму. У хорошому стані вакуумна трубка має сріблястий колір. Якщо він виглядає білим, це означає, що вакуум зник і трубку потрібно замінити.

Вакуумний колектор складається з комплексу вакуумних трубок (10-30) і здійснює передачу тепла в резервуар для зберігання через рідину. Ефективність вакуумних колекторів висока:

- при похмурій погоді, тому що вакуумні трубки можуть поглинати енергію інфрачервоних променів, які проходять через хмари;
- можуть працювати при мінусових температурах.

Рисунок 3.3- Вакумний сонячний колектор

3.4 Сонячні батареї

Сонячна батарея - це набір модулів, що сприймають і перетворюють сонячну енергію, в тому числі і теплових(рисунок 3.4.1). Але цей термін традиційно закріпився за фітоелектричними перетворювачами. Тому, кажучи

«сонячна батарея» маємо на увазі фітоелектричний пристрій, що перетворює сонячну енергію в електричну.

Сонячні батареї здатні генерувати електричну енергію постійно або акумулювати її для подальшого використання. Вперше фотоелектричні батареї були застосовані в на космічних супутниках.

Гідність сонячних батарей - максимальна простота конструкції, простий монтаж, мінімальні вимоги до обслуговування, великий термін експлуатації. При установці не вимагають додаткового місця. Єдине умова - не затінювати їх протягом тривалого часу і видаляти пил з робочої поверхні. Сучасні сонячні батареї здатні зберігати працездатність протягом десятиліть! Важко знайти систему настільки безпечну, ефективну і з таким тривалим терміном дії! Вони виробляють енергію протягом усього світлового дня, навіть у похмуру погоду.

Рисунок 3.4- Сонячна батарея

Сонячні батареї мають свої недоліки в застосуванні:

- чутливість до забруднень. (Якщо розташувати батарею під кутом 45 градусів, то вона буде очищена дощами або снігом, тим самим не буде потрібно додаткового обслуговування)
- чутливість до високої температури. (Так, при нагріванні до 100 - 125 градусів сонячна батарея може навіть відключитися і може знадобитися

система охолодження. Вентиляційна система при цьому витратить малу частку вироблюваної батареєю енергії. В сучасних конструкціях сонячних батарей передбачена система відтоку гарячого повітря.)

- висока ціна. Але беручи до уваги тривалий термін служби сонячних батарей, можна зрозуміти, що вона не тільки окупить витрати на її придбання, але і заощадить кошти при споживанні електроенергії, заощадить тони традиційних видів палива при тому екологічно безпечна.

3.5 Види сонячних батарей

Найбільш поширеним і популярним типом сонячних батарей є монокристалічні кремнієві сонячні елементи (рисунок 3.5.1). Їх отримують литтям кристалів кремнію високої чистоти, в яких розплав застигає при контакті з затравочним кристалом. У процесі охолодження кремній поступово зміцнюється у вигляді циліндричного лиття монокристала діаметром 13–20 см, довжина якого досягає 200 см. Отриманий таким чином злиток розрізають на листи товщиною 250–300 мкм. Такі елементи мають більш високу ефективність порівняно з елементами, виробленими іншими методами, ККД досягає 19%, внаслідок особливої орієнтації монокристалічних атомів, що сприяє зростанню рухливості електронів. Кремній проникає в сітку металевих електродів. Традиційно монокристалічні модулі вставляються в алюмінієвий каркас і покриваються протиударним склом. Колір монокристалічних фотоелементів темно-синій або чорний.

Панелі сонячних батарей надійні, довговічні (термін служби до 50 років) і прості в установці, оскільки не містять рухомих частин. Сонячні батареї можна використовувати там, де нормальне живлення і велика кількість сонячних днів не працюють добре. Приклади сонячних панелей: на дахах будинків для виробництва електроенергії, на вуличних і садових лампах для освітлення,

підзарядці акумуляторів, забезпечення обладнанням для суден, радіостанцій, насосів, сигналізацій тощо.

Рисунок 3.5- Монокристалічна сонячна панель

Сонячні батареї, виготовлені з монокристалічних фотоелектричних елементів, є більш ефективними, але й більш дорогими з точки зору ват потужності. Їх ефективність зазвичай знаходиться в межах 14-18%.

Як правило, монокристалічні елементи мають форму полігонів, з якими важко заповнити всю площу панелі без залишку. В результаті, щільність потужності сонячного елемента трохи нижче, ніж щільність потужності його окремого елемента.

Інший вид сонячних панелей– багатокристалічні кремнієві сонячні елементи(рисунок 3.6). Виготовлення мультикристалічного кремнію значно простіше, ніж монокристалічного. Мультикристалічний кремній, як матеріал складається з випадково зібраних різних монокристалічних кремнієвих граток (термін служби 25 років, ККД до 15%). Тому багатокристалічні панелі зазвичай пропонуються дешевше.

Рисунок 3.6- Багатокристалічна сонячна панель

Полікристалічні кремнієві сонячні панелі э альтернативою монокристалічним панелям (рисунок 3.7). Полікристалічний кремній має нижчу собівартість. Кристали в ньому все ще є агрегатними, але мають різну форму і орієнтацію. Цей матеріал, в порівнянні з темними монокристалами, відрізняється яскраво-синім кольором. Удосконалення процесу виробництва елементів даного типу дозволяє отримати сьогодні компоненти, характеристики яких лише трохи поступаються за електричними характеристиками монокристалу.

За допомогою сонячної системи ви можете:

- освітлення та постачання електроенергії будинками та котеджами, школами, лікарнями, офісами, фермами, теплицями тощо;
- освітлення парків, садів, дворів, автомагістралей і вулиць;
- забезпечення електрозв'язку, медичного обладнання;

- постачання енергоносіїв на нафто- і газопроводи;
- забезпечити електропостачання та опріснення;
- заряджати мобільні телефони та ноутбуки.

Рисунок 3.7- Полікристалічна сонячна панель

Тонкоплівкові батареї дозволяють виготовити більш дешеву панель за ціною виробництва. Ця обставина робить плівкові плівки більш привабливими для будівництва великих «ферм» для виробництва електроенергії від сонячного світла, коли «сонячний фермер» обмежений не стільки землею, скільки ціною встановлення батареї. Вона може бути встановлена не тільки на даху, але і на бічних поверхнях будівлі.

Тонкоплівкові панелі не потребують прямих сонячних променів, вони працюють з дифузним випромінюванням, завдяки чому загальна потужність, вироблена протягом року, на 10-15% більше, ніж виробляються традиційні кристалічні сонячні панелі. Тонка плівка є набагато більш рентабельним способом отримання енергії і може вигравати монокристали в місцях з

туманним, похмурим кліматом або в тих галузях, які характеризуються пильністю повітря або високим вмістом інших твердих частинок.

У 95% випадків тонкоплівкові панелі використовуються для мережевих систем, які виробляють електроенергію безпосередньо в мережі. Для цих панелей необхідно використовувати високовольтні контролери та інвертори, які не зв'язані з малопотужними побутовими системами.

Хоча вартість тонкоплівкових панелей невелика, вони займають значно більшу площу (в 2,5 рази), ніж моно- і полікристалічні панелі. Через зниження ефективності. Тонкоплівкові панелі можна ефективно використовувати в системах потужністю 10 кВт і більше. Для побудови невеликих автономних або резервних систем електроживлення використовувалися монокристалічні і полікристалічні панелі.

Сонячні панелі телуриду кадмію (CdTe) засновані на технології плівки. Напівпровідниковий шар наносять тонким шаром у кілька сотень мікрометрів. Ефективність елементів з телуриду кадмію невелика, ефективність - близько 11%. Однак у порівнянні з кремнієвими панелями, вати цих батарей коштують на кілька десятків відсотків дешевше.

Рисунок 3.8- Сонячна панель із телуриду камдію

Панелі сонячних батарей на базі CIGS (рисунок 3.9). CIGS являє собою напівпровідник, що складається з міді, галію і селену. Цей тип сонячних

елементів також виготовлений за технологією плівки, але в порівнянні з панелями телуриду кадмію він має більш високу ефективність, ефективність досягає 15%.

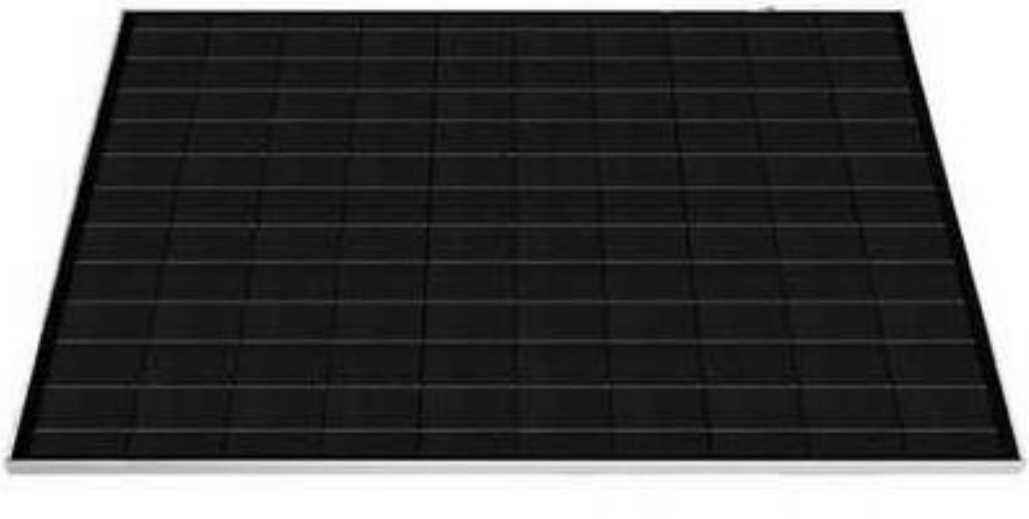


Рисунок 3.9- Панелі сонячних батарей на базі CIGS

Потенційні покупці сонячних батарей часто задаються питанням, чи може той чи інший тип фотоелектричних перетворювачів забезпечити необхідну потужність всієї системи. Тут треба розуміти, що ефективність сонячних елементів безпосередньо не впливає на кількість енергії, що виробляється установкою.

Така ж потужність всієї установки може бути отримана за допомогою будь-якого типу сонячних елементів, але більш ефективні фотоелектричні перетворювачі займатимуть менше місця і вимагатимуть меншої площі для їх розміщення. Наприклад, якщо потрібно близько 8 квадратних метрів, щоб отримати один кіловат електроенергії, то поверхня сонячного елемента базується на монокристалічному кремнії, тоді аморфні кремнієві панелі займуть близько 20 кв. Наведений приклад, звичайно, не є абсолютним. На виробництво електроенергії фотоелектричними камерами впливає не тільки загальна площа сонячних батарей. Електричні параметри будь-якої сонячної батареї визначаються в так званих стандартних умовах випробувань, а саме, коли інтенсивність сонячного випромінювання становить 1000 Вт / кв.м. і робочу температуру панелі 25 ° С.

У країнах Центральної та Східної Європи інтенсивність сонячного випромінювання рідко досягає номінальної вартості, тому навіть у сонячні дні фотоелектричні панелі працюють з недовантаженням. Може здатися, що температура 25 ° С теж не така поширена. Однак мова йде про температуру сонячної панелі, а не про температуру повітря.

У рамках загальної тенденції зниження вихідної потужності при збільшенні робочої температури кожен тип сонячних елементів веде себе по-різному. Таким чином, в кремнієвих осередках номінальна потужність падає з кожним ступенем перевищення номінальної температури на 0,43-0,47%. В той же час, елементи телуриду кадмію втрачають лише 0,25%.

4. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ АНАЛІЗУ СОНЯЧНОЇ АКТИВНОСТІ

З кожним роком сонячна енергетика набуває все більшої популярності. Для аналізу вигідності використання цього джерела енергії вже побудовано багато різноманітного програмного забезпечення, яке вирішує дуже широкий спектр прикладних задач. Серед них можна виділити і задачу аналізу сонячної активності в різних регіонах.

З метою дослідження та розрахунку прибутковості сонячних електростанцій і геліосистем було запропоновано розробити програмну систему, яка продемонструє сонячну активність в регіонах та зможе за індивідуальними даними розрахувати помісячного видобутку електроенергії за допомогою сонячних панелей та теплої води за допомогою геліосистем.

4.1 Аналіз існуючих програмних засобів

У процесі пошуку інформації, та аналізу існуючих рішень, було виявлено, що на даний момент подібні системи є досить складними у використанні, або реалізують поставлену задачу не в повному обсязі:

— “solar-battery.com.ua” — це веб-сайт, який надає дані сонячної активності по різним областям України. Недоліком даної системи перед поставленою задачею є те, що вона не має опції розрахунку прибутку від сонячних систем та геліосистем. Крім того карта не є інтерактивною, тобто користувач не може з нею взаємодіяти;

— “atmosfera.ua” — це веб-сайт, в якому є калькулятор сонячної електростанції, в якому можна обрати регіон країни та потужність електростанції. Недоліком є те, що неможна вибирати тип сонячних панелей і не зрозуміло для якого типу проводиться обчислення. Також на сайті нема можливості зробити обчислення для геліосистем.

— “enertime.com.ua” — це веб-сайт, в якому є калькулятор сонячної електростанції, в якому можна обрати потужність та тип електростанції, але не можна вибрати регіон країни.

Не у всіх вже існуючих систем є можливість побачити дані по видобутку електроенергії помісячно і не завжди в зручному форматі. Вище приведені сайти використовують лише сонячні батареї і не дають змогу зробити розрахунок для геліосистем. Багато цих рішень є корпоративними, які не дають великий вибір різних типів станцій, але використовують лише свої продукти для розрахунку.

5. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Аналізуючи поставлену задачу та методи її вирішення, було вирішено втілювати рішення в якості веб-додатку. Головною перевагою веб-додатку є його універсальність і можливість використання на будь-яких пристроях та операційних системах за допомогою бразурера.

5.1 Вибір архітектури програмного комплексу

Для реалізації поставленої задачі було вирішено використовувати триланкову архітектуру, яка складається з таких компонентів: сервер, база даних і клієнт. Схема даної архітектури зображена на рисунку 3.1.

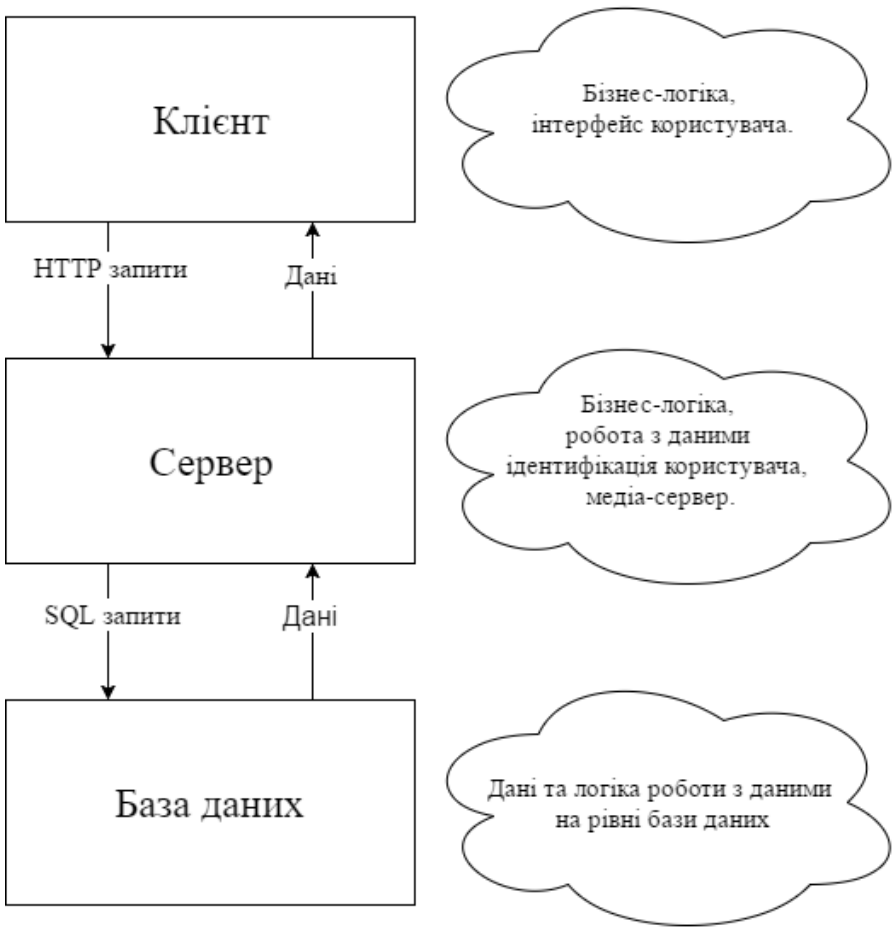


Рисунок 5.1 – Триланкова архітектура програмного комплексу

Головним центром системи є сервер. У ньому зосереджена доступу до бази даних. За допомогою серверу відбувається ідентифікація адміністратора для надання індивідуального доступу до налаштувань програмного застосунку. Звичайний користувач може користуватись програмним забезпеченням без реєстрації. Користувач може звертатись до бази даних тільки через сервер, щоб уникнути можливості пошкодження даних та їх використання не за призначенням.

Під час користування програмою, користувач взаємодіє з клієнтським додатком, яким в даному випадку є веб-сайтом. Користувачу пропонується веб-інтерфейс, за допомогою якого відбувається взаємодія з картою сонячної радіації та калькуляторам підрахунку прибутку.

Головною задачею рівня бази даних є забезпечення збереження даних, які сервер зберігає для подальшого використання. Крім того, забезпечується цілісність даних за допомогою зовнішніх звя’зків та ключів. На рівні бази даних також можна реалізовувати деяку бізнес-логіку, яка не потребує використання зовнішніх джерел даних окрім самої бази даних та її таблиць.

5.2 Опис архітектури серверу

Шаблон проектування — це архітектура, рішення яке описує яким чином вирішуються задачі, які часто зустрічаються при розробці програмних систем чи додатків.

Для реалізації серверу було використано фреймворк, який реалізує шаблону проектування MVC [1]. Патерн MVC (рисунок 5.2)- це архітектурний шаблон, що часто використовується для розробки призначених для користувача інтерфейсів, який ділить додаток на три взаємопов'язані частини. Це робиться для того, щоб відокремити внутрішнє представлення інформації від того, як інформація може надаватися і приймається від користувача. Шаблон проектування MVC розділяє ці основні компоненти, дозволяючи повторне використання коду і паралельну розробку.

Трьома основними компонентами MVC э:

- M — Model (Модель).
- V — View (Представлення).
- C — Controller (Контролер).

Таким чином, редагування будь-якого компонента може відбуватися незалежно.

Модель- центральний компонент візерунка. Це динамічна структура даних програми, незалежна від призначеного для користувача інтерфейсу. Він безпосередньо керує даними, логікою і правилами програми.

Представлення- будь-яке уявлення інформації, такої як діаграма або таблиця. Можливі кілька видів однієї і тієї ж інформації, наприклад: гістограма для управління і табличне представлення для бухгалтерів.

Контролер- приймає запит і перетворює його в команди для моделі або представлення. Крім поділу на компоненти, шаблон модель-представлення-контролер визначає взаємодію між ними.

Модель відповідає за управління даними додатка. Вона отримує значення для користувача від контролера. Представлення означає уявлення моделі в певному форматі. Контролер реагує на введення користувача і виконує взаємодії з об'єктами моделі даних. Контролер отримує вхідні дані, при необхідності перевіряє їх і потім передає вхідні дані в модель.

Головними цілями MVC э створення умов для паралельної розробки та можливості повторного використання коду. У цілому MVC поділяє різні компоненти додатки, розробники можуть працювати паралельно над різними компонентами, не впливаючи і не блокуючи іншу. Наприклад, команда може розділити своїх розробників між інтерфейсом і серверною частиною. Back-end розробники можуть займатися структуризацією даних і того, як користувач взаємодіє з ними і не турбуватись про інтерфейс користувача. І наоборот, розробники інтерфейсу можуть спростити і протестувати макет додатків до того, як структура даних стане доступною.

Також, одне і те саме (або подібне) представлення для одного додатка може бути реорганізовано для іншого додатка з іншими даними, оскільки представлення просто обробляє здатність відображення даних для користувача. На жаль, це не працює, коли цей код також використовується для обробки користувацького вікна. Наприклад, код DOM (включаючи користувальницькі абстракції з додатками) потрібен як для графічного відображення, так і для вводу інформації користувача. (Зверніть увагу, що, незважаючи на назву Document Object Model, DOM на справді не є моделлю MVC, тому що це інтерфейс користувача).

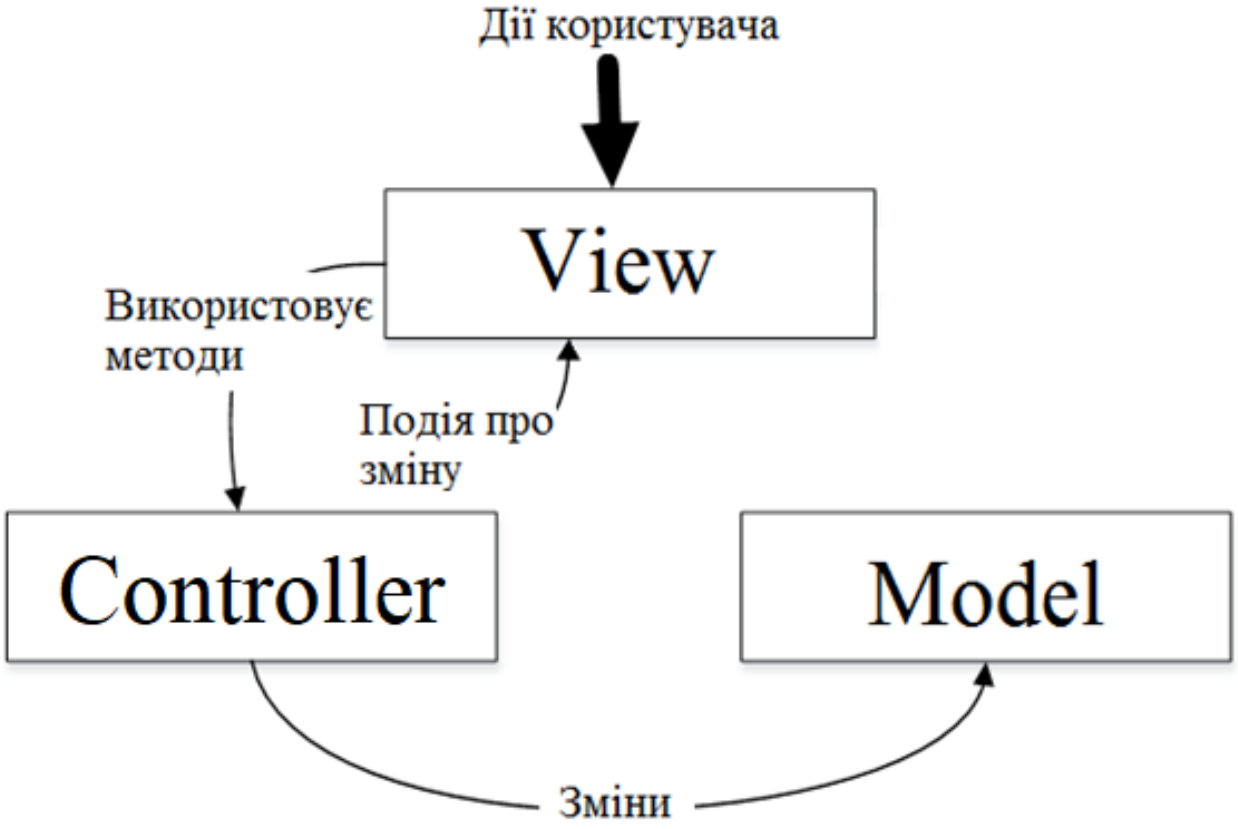


Рисунок 5.2 – Схема роботи MVC шаблону

Для вирішення цих проблем MVC (і подібні йому шаблони) часто об'єднують з компонентною архітектурою, яка надає набір елементів користувальницького інтерфейсу. Один з основних функцій користувальницького інтерфейсу являє собою один з компонентів більш

високого рівня, який об'єднав 3 необхідних компонента MVC в один пакет. Створення таких компонентів більш високого рівня, які не залежать від інших, розробники можуть швидко і легко повторно використовувати компоненти в інших додатках.

Як і кожен шаблон, MVC має свої недоліки та переваги.

Перевагами є:

- синхронна розробка- кілька розробників можуть працювати одночасно на моделі, контролері і переглядах;
- висока згуртованість- MVC дозволяє логічне групування відповідних дій на контролері разом. Погляди для конкретної моделі також згруповані разом;
- loose coupling- сама природа рамки MVC така, що існує низька зв'язок між моделями, поглядами або контролерами;
- легкість модифікації- через поділ обов'язків, майбутній розвиток та модифікація проводяться простіше;
- кілька представлень для моделі - моделі можуть мати декілька представлень.

Недоліки MVC можна в цілому класифікувати як накладні витрати на некоректно розроблене програмне забезпечення. Наприклад:

- Навігація за кодами - навігація фреймворків може бути складною, оскільки вона запроваджує нові рівні абстракції і вимагає, щоб користувачі адаптувалися до критеріїв декомпозиції MVC.
- послідовність мульти артефактів - розкладання функції на три артефакти викликає розсіювання. Таким чином, вимагають від розробників збереження узгодженості декількох подань одночасно;
- підірвані неминучою кластеризацією - додатки, як правило, мають важку взаємодію між тим, що бачить користувач і тим, що користувач використовує. Тому обчислення кожної функції і стан, як правило,

скомпоновані в одну з 3 частин програми, видаляючи передбачувані переваги MVC;

- надмірний шаблон - через те, що обчислення та стан додатків зазвичай групуються в одну з трьох частин, інші частини вироджуються або в корінні шаблони, або за кодом, що існує лише для задоволення шаблону MVC;
- висвітлена крива навчання- знання про множинні технології стає нормою. Розробники, що використовують MVC, повинні бути кваліфікованими в декількох технологіях;
- відсутність додаткових переваг- додатки користувальницького інтерфейсу вже враховані в компонентах і досягають повторного використання коду та незалежності через архітектуру компонентів, не залишаючи додаткових переваг для MVC.

5.3 Опис архітектури клієнтського застосунку

В якості клієнтського застосунку буде виступати веб-інтерфейс розроблений з використанням layout view in ASP.NET Core MVC.

Програма може містити загальні частини інтерфейсу користувача, які залишаються однаковими по всьому застосуванню, наприклад, логотип, заголовок, ліва навігаційна панель, правий або нижній колонтитул. Для вирішення проблеми дублювання коду ASP.NET Core MVC має view Layout, який містить ці загальні частини інтерфейсу, так що нам не потрібно писати один і той же код на кожній сторінці. Вікно компонування є таким же, як і головна сторінка веб-додатку. Вигляд макета дозволяє визначити загальний шаблон сайту, який можна успадковувати в декількох поданнях, щоб забезпечити узгоджений вигляд і відчуття на декількох сторінках програми. View макет усуває дублювання кодування і підвищує швидкість розробки, простий в обслуговуванні.

5.4 Опис інструментів розробки

Програмний комплекс побудований за принципами триланкової архітектури побудови програм. Кожен рівень цієї архітектури реалізовано з використанням різних технологій і головною ціллю було створення мультиплатформного рішення з використанням відкритих технологій.

Для клієнтського рівня було використано такий набір технологій: html, css для створення макету та його стиліцзації, Razor для сервеного рендерингу html коду, javascript для динаміки веб-додатку, фреймворк для створення графічних інтерфейсів Boostrap, js бібліотека leaflet для відображення інтерактивної карти, Plotly.js для побудови діаграм.

HTML- це комп'ютерна мова розмітки, створена для розробки веб-сайтів. Ці веб-сайти можуть переглядати будь-хто, хто підключився до інтернету. Визначення HTML- це мова для розмітки HYPERTEXT. HyperText - це метод, за допомогою якого ви пересуваєтеся в інтернеті - клацаючи на спеціальному тексті, який називається гіперпосиланням, яке переносять вас на наступну сторінку. Той факт, що він гіпер, означає, що він не є лінійним- тобто ви можете перейти в будь-яке місце в інтернеті, коли ви хочете, натиснувши на посилання. Розмітка- це те, що роблять HTML-теги до тексту всередині них. Вони позначають його як певний тип тексту (наприклад, курсивний текст) і браузер розрізняє ці позначення та відображає текст належним чином.

CSS- це мова опису зовнішнього виду документа, написаного з використанням HTML. Використовується творцями веб-сторінок для завдання кольорів, шрифтів, правильного розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих веб-сторінок. Основною метою розробки CSS було розділення опису логічної структури веб-сторінки (яке проводиться за допомогою HTML) від опису зовнішнього виду веб-сторінки.

Javascript (JS) - це скриптова мова, яка в основному використовуються в веб розрозробках. Вона використовується для поліпшення HTML-сторінок і зазвичай знаходиться в HTML-коді. JavaScript є інтерпретованою мовою. Таким

чином, її не потрібно компілювати. JavaScript надає веб-сторінки інтерактивності та динамічності. Це дозволяє сторінкам реагувати на події, демонструвати спеціальні ефекти, приймати змінний текст, перевіряти дані, створювати файли cookie, виявляти тип браузера користувача тощо.

Інструмент Razor - це синтаксис розмітки, який дозволяє вставляти серверний код у веб-сторінки за допомогою C # і VB.Net. Це не мова програмування. Це мова розмітки на стороні сервера. Мова, яку Razor використовує для розмітки, за замовчуванням - це HTML. Візуалізація HTML з розмітки Razor не відрізняється від рендеринга HTML з файлу HTML. Razor використовує символ @ для переходу з HTML на C #. Razor аналізує вирази C # і передає їх у вихідний HTML.

Бібліотека Bootstrap — це набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Бібліотека Leaflet — це JavaScript бібліотека з відкритим кодом ядра для відображення карт на web-сторінках. Вона підтримує сучасні фреймоврки, не велика за обсягом та зручна у використані. Бібліотека реалізує підтримку шарів мап, які побудовані за технологією WMS, підтримує GeoJSON та має здатність відображати поверхню в векторному виді. Інші типи проекцій мап можуть підтримуватись за допомогою додатків. Головне, щоб кожний формат мав однозначну функцію перетворення локальних координат в географічні координати відповідно до картографічної проекції, в якої ці локальні координати завдані.

Бібліотека Leaflet не потребує від користувача бути досвідченим в картографічних сервісах, що значно спрощує задачу вбудови мапи в html-сторінки та веб-додатки. Крім того, бібліотека дозволяє працювати із різними слоями, а в якості джерел мапи можна використовувати будь-якій публічний веб-сервіс тайлів (порізані зображення мапи).

Провайдер OpenStreetMap (відкрита вулична мапа) — це відкритий проект цілями якого є збір, збереження та розповсюдження загальнодоступних геопросторових даних, а також створення інструментів для роботи з ними.

Бібліотека Plotly.js — це відкрита JavaScript бібліотека, яка використовується для побудови різноманітних графіків та діаграм. Вона представляє високорівневий доступ до інших бібліотек, що дає доволі зручний підхід для візуалізації даних.

Для серверного рівня було використано такі технології: мова програмування C# та платформа .NET Core, фреймворк для створення RESTful сервісів ASP.NET Core MVC, ORM для доступу до бази даних Entity Framework Core.

Мова C # - це об'єктно-орієнтована мова програмування. Розроблена в 1998–2001 роках групою інженерів Microsoft, під керівництвом Андерса Хейлсберга і Скотта Вілтомута , як мова для розробки додатків для Microsoft .NET Framework. Згодом він був стандартизований як ECMA-334 і ISO / IEC 23270.

Мова C # відноситься до сімейства мов з C-подібним синтаксисом, його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження оператора (включаючи явні та неявні перетворення типу), делегатів, атрибутів, подій, властивостей, загальних типів і методів, ітераторів, анонімних функцій з закриттями, LINQ, виключень, коментарів у форматі XML.

Зайнявши чимало від своїх попередників - C ++, Pascal, Modula, Smalltalk і, зокрема, Java - C #, спираючись на практику їх використання, виключає деякі моделі, які виявилися проблематичними при розробці програмних систем. Наприклад, C #, на відміну від C ++ і деяких інших мов, не підтримує успадкування декількох класів (тим часом дозволено успадкування декількох інтерфейсів). [1].

Мова C # була розроблена, як мова програмування на рівні додатків для CLR і, як така, залежить, перш за все, від можливостей самого CLR. Це

стосується, в першу чергу, системи типу C #, яка відображає BCL. Наявність або відсутність деяких експресивних ознак мови диктується тим, чи може конкретна мовна особливість бути переведена у відповідні CLR-конструкції. Таким чином, з розвитком CLR від версії 1.1 до 2.0, C # сама була значно збагачена; Таку взаємодію слід очікувати в майбутньому (однак, ця модель була порушена з виходом C # 3.0, який є розширенням мови, які не покладаються на розширення .NET). CLR надає C #, як і всі інші .NET-орієнтовані мови, безліч функцій, яких «класичні» мови програмування бракують. Наприклад, збирання сміття в C # не реалізовано, але CLR робиться для програм, написаних на C #, як це робиться для програм на VB.NET, J # та інших.

Фреймворк .NET Core базується на .NET Framework. Платформа .NET Core відрізняється від неї модульністю, крос-платформним, можливістю використання хмарних технологій, а також тим, що між бібліотекою CoreFX і середовищем виконання CoreCLR існує поділ.

.NET Core - модульна платформа. Кожен компонент оновлюється за допомогою менеджера пакетів NuGet, що означає, що він може оновлювати свої модулі окремо, тоді як .NET Framework оновлюється тільки повністю. Кожна програма може працювати з різними модулями і не залежить від єдиного оновлення платформи.

CoreFX - це бібліотека, інтегрована в .NET Core. Серед її складових: System.Collections, System.IO System.Xml .

CoreCLR є середовищем виконання, що включає в себе компілятор RyuJIT (JIT), вбудований збирач сміття та інші компоненти.

Фреймворк ASP.NET Core - це кросплатформений фреймворк, для створення сучасних хмарних додатків, підключених до Інтернету. Фреймворк ASP.NET Core довзволяє:

- створення веб-додатків і служб, додатків IoT і мобільних back-end частин;
- використання ваших улюблених інструментів розробки для Windows, MacOS і Linux;

- розгортання в хмарі або локально;
- запуск як на .NET Core так і .NET Framework.

ASP.NET Core - це редизайн ASP.NET 4.x з архітектурними змінами, які призводять до більш компактною і модульній структурі. ASP.NET Core надає наступні переваги:

- має єдину історію створення веб-інтерфейсу і веб-API;
- зручний у тестуванні;
- інструмент Razor Pages робить розробку орієнтованих на сторінки сценаріїв простішим і продуктивним;
  blazor дозволяє використовувати C # в браузері разом з JavaScript;
- спільне використання логіки на стороні сервера і на стороні клієнта, написаної на .NET;
- можливість розробки і запуску на Windows, MacOS і Linux;
- відкритий вихідний код;
- інтеграція сучасних клієнтських середовищ і робочих процесів розробки;
- готова до роботи в хмарі система конфігурації на основі середовища;
- вбудована ін'єкція залежностей;
- легкий, високопродуктивний і модульний конвеєр HTTP-запитів;
- можливість розміщення на IIS, Nginx, Apache, Docker або самостійного розміщення на своєму власному процесі;
- інструмент, який спрощує сучасну веб-розробку.

Entity Framework - це спеціальна об'єктно-орієнтована технологія, заснована на .NET framework для роботи з даними. Якщо традиційні інструменти ADO.NET дозволяють створювати з'єднання, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework є більш високим рівнем абстракції, що дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. . Якщо на фізичному рівні ми працюємо з таблицями, індексами, первинними і зовнішніми ключами, то на

концептуальному рівні, що пропонує нам Entity Framework, ми вже працюємо з об'єктами.

Перша версія Entity Framework - 1.0 була випущена ще в 2008 році і представлена дуже обмеженою функціональністю, базовою підтримкою ORM (об'єктно-реляційне відображення - відображення даних про реальні об'єкти) і єдиним підходом до взаємодії з базою даних - Database First. З виходом версії 4.0 у 2010 році багато чого змінилося - з тих пір Entity Framework став рекомендованою технологією доступу до даних, а нові можливості взаємодії з підходами Model First і Code First були введені в рамки.

Додаткові поліпшення функціональності відбулися з виходом версії 5.0 в 2012 році. Нарешті, в 2013 році було випущено Entity Framework 6.0, з можливістю асинхронного доступу до даних.

Основна концепція Entity Framework є поняття сутності або entity. Сутність являє собою набір даних, пов'язаних з певним об'єктом. Тому ця технологія передбачає роботу не з таблицями, а з об'єктами та їх множинами.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має ряд властивостей. Наприклад, якщо сутність описує людину, то можна виділити такі властивості, як ім'я, прізвище, висота, вік, вага. Властивості не обов'язково являють собою прості дані типу int, але можуть також представляти більш складні структури даних. І кожен об'єкт може мати одну або кілька властивостей, які будуть відрізняти цю сутність від інших і однозначно визначатиме цю сутність. Ці властивості називаються ключами.

У цьому випадку об'єднання можуть бути з'єднані асоціативністю "один до багатьох", "один-до-одного" та "багато-до-багатьох", як і в реальній базі даних, спілкування відбувається через зовнішні ключі.

Відмінною особливістю Entity Framework є використання запитів LINQ для вибірки даних з бази даних. За допомогою LINQ ми можемо не тільки витягти певні рядки, які зберігають об'єкти з бази даних, але й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншою ключовою концепцією є модель даних суб'єкта. Ця модель порівнює класи об'єктів з реальними таблицями в базі даних.

Базою даних було обрано PostgreSQL. База даних PostgreSQL — це об'єктно-реляційна система керування базами даних. Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не належить одній компанії, її розробка проводиться завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення [8].

5.5 Обґрунтування вибору програмної реалізації

При проектуванні системи було вивчено та проаналізовано предметну область та вимоги замовника. Після ретельного аналізу було вирішено розроблювати програмний продукт, який заснований на веб-технологіях для використання за допомогою веб-браузера.

Технології, які використовуються на сервері, були обрані за принципом зручності у використанні, відкритості вихідних кодів, актуальності в наш час та можливістю виконання на будь-якій операційній системі. Платформа .NET Core надає змогу створювати програми на мовах сімейства .NET, зокрема мова C#, на будь-якій операційній системі. Фреймворк ASP.NET Core MVC надає величезні можливості для створення веб-серверів будь-якої складності, забезпечуючи при цьому достатню швидкість роботи та надійність. Також перевагою використання цих технологій є те, що їх можна запустити у Docker контейнерах та побудувати на цьому швидке та надійне розгортання серверів з подальшим масштабуванням без створення при цьому спеціальної архітектури програмного забезпечення. Для доступу до бази даних було використано фреймворк Entity Framework Core через його зручність та універсальність. За допомогою цього фреймворку можливо абстрагуватися від конкретної реалізації бази даних і використати відповідні моделі даних для роботи з даними. Це надає змогу зосередити всю увагу на створенні якісної та протестованої бізнес-логіки.

На клієнтському рівні було обрано технології, які прості у використанні та швидкі при роботі. Razor було обрано через те, що за допомогою нього можливо створювати складні графічні інтерфейси, які легко модифікувати, тестувати та розширювати в подальших циклах розробки програмного забезпечення використовуючи серверний рендеринг мовою C#. Мова Javascript була обрана через те, що вона підтримує безліч модулів та є однією з найпопулярніших для розробки клієнтської частини. Також ця мова програмування надає безліч різноманітного “синтаксичного цукру”, який дозволяє скоротити кількість зайвого коду. Для того щоб стилізувати клієнтський додаток було вирішено використати фреймворк Bootstrap через його багату бібліотеку готових інтерфейсів та вбудовану підтримку та масштабування до розмірів мобільного телефону. Відображення карти та взаємодія з нею реалізована за допомогою Leaflet. Бібліотеку Leaflet було через те, що вона є відкрітою та зручною, дозволяє робити велики набір операцій з картою. За допомогою цієї бібліотеки можна додавати нові маркери на карті та виділяти потрібні регіони за допомогою формату GeoJson. GeoJson – це стандарт, який використовується для виділення окремих регіонів, областей, країн та ін. на картах. Було отримано дані для областей України, що дає змогу зв’язати їх з даними по сонячній радіації в Україні по регіонах. Ця бібліотека може працювати с великою кількістю карт. Одна з таких — це OpenStreetMap. OpenStreetMap була обрана тому, що це відкрита карта, яка охоплює весь світ. Це безкоштовна карта, дані в цій карті постійно оновлюються завдяки великої кількості волонтерів, які постійно її оновлюють та займаються її розробкою.

Plotly.js був обраний завдяки простому інтерфейсу взаємодії з фреймворком та можливістю конфігурувати діаграми будь-яким способом.

Ця технологія дуже потрібна, якщо веб-сайт планується запускати на віддаленому сервері.

Базою даних було обрано PostgreSQL через те, що це проект з відкритим вихідним кодом і надзвичайно великою підтримою з боку розробників. Також важливою властивістю є велика швидкість роботи, надійність та кількість

вбудованих можливостей. Цю базу даних можливо розгорнути на будь-якій сучасній операційній системі або в Docker контейнері.

Дані технології в сукупності дають змогу збудувати якісний та надійний продукт, який захищений від патентних позовів з боку розробників, бо всі ці технології покриті ліцензіями, які виключають таку можливість і надають доступ до вихідних кодів даних проектів.

6. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Застосунок буде складатись з веб-сторінки, де користувачеві буде доступна інтерактивна карта та панель обрахунку. Користувач зможе поставити маркер на карті, створити новий або вибрати вже існуючий. Крім того, користувач може редагувати інформацію, що належить кожному маркеру. Усі елементи будуть знаходитись на боковій панелі. Такий підхід зробить інтерфейс максимально інтуїтивно зрозумілим та простим.

На рисунку 4.1 наведена схема структури системи

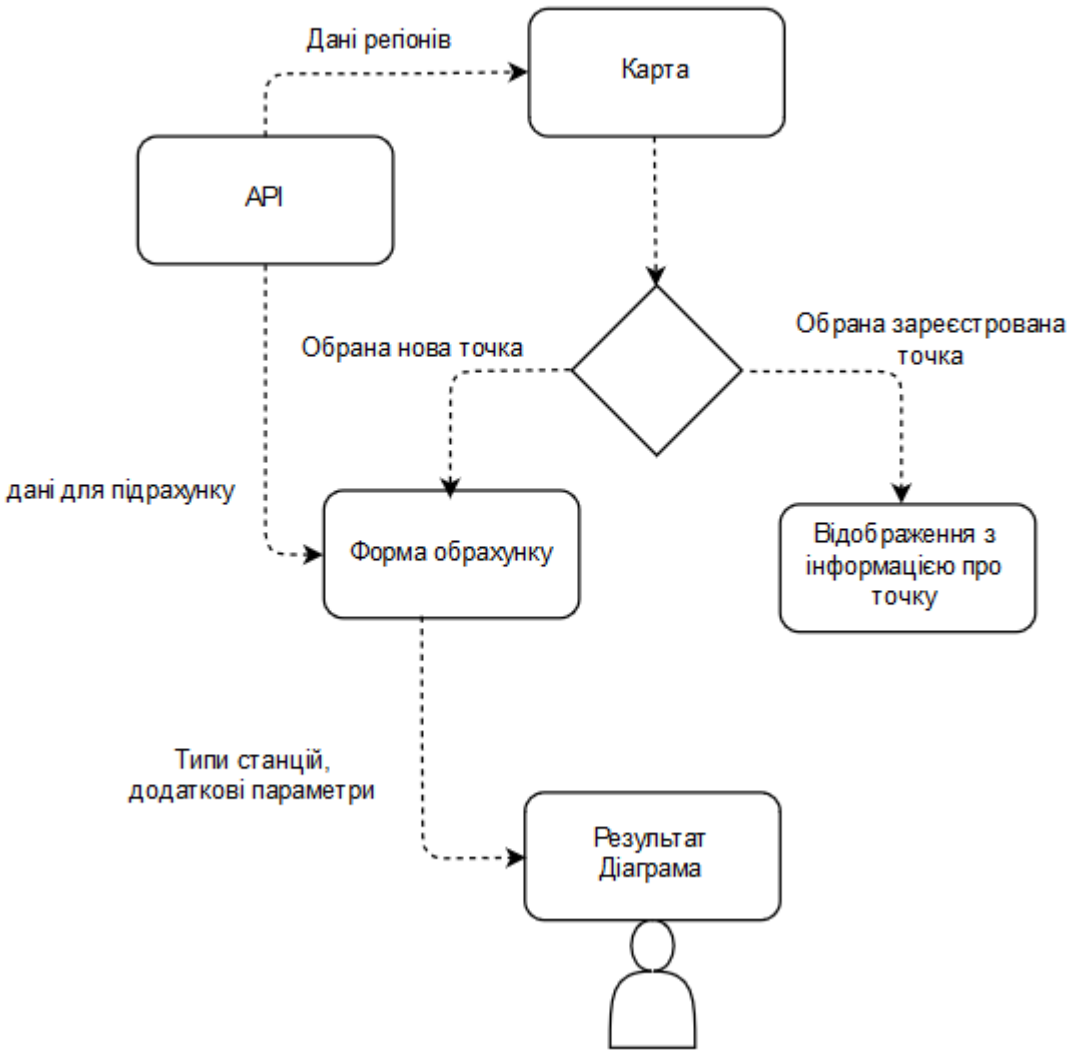


Рисунок 6.1 — Схема структури системи

6.1 Опис функціональності системи

Додаток для підрахунку об’єму видобутку електроенергії або гарячої води за допомогою сонця містить має лише одного актора — користувач, який зайшов на веб-сторінку

На рисунку 4.2 представлена діаграма прецедентів, яка описує функції та дії актора у системі.

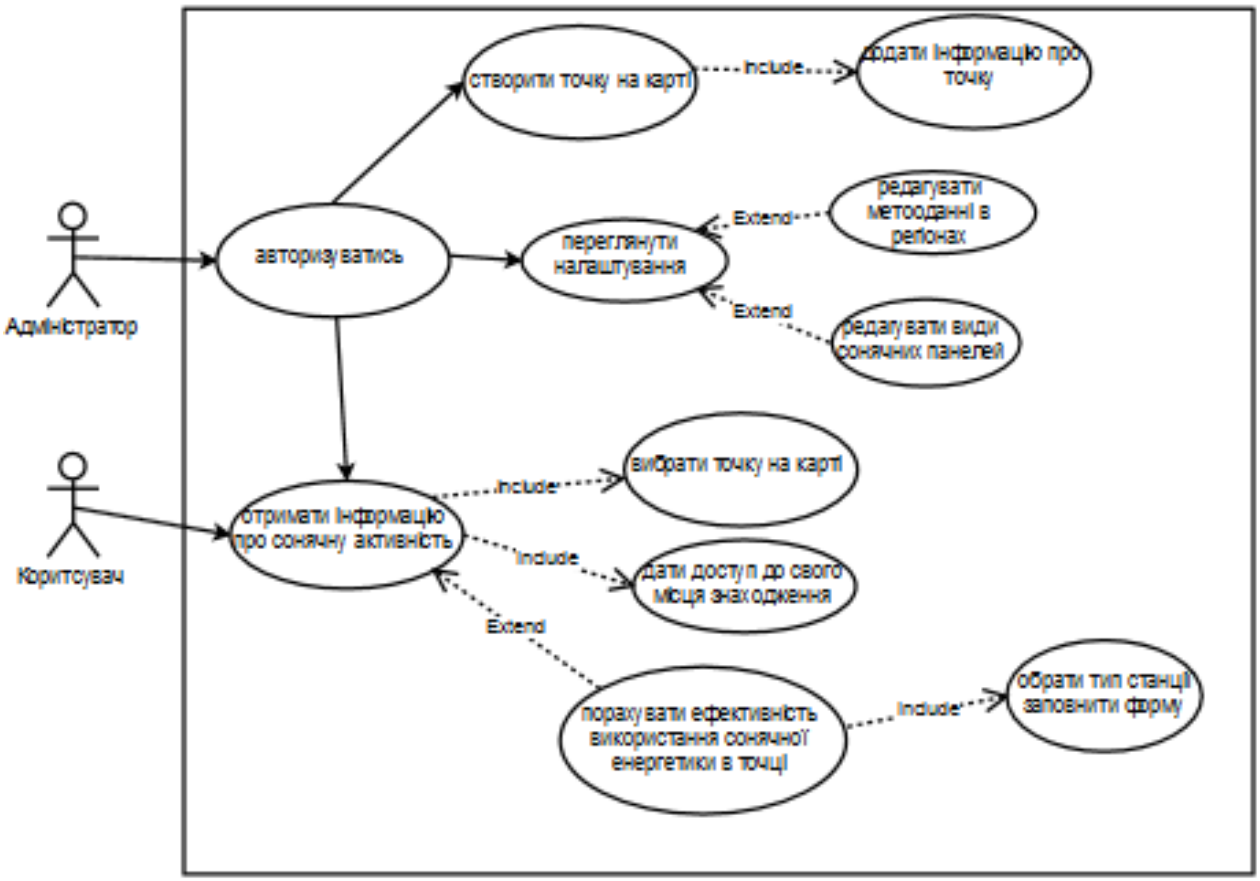


Рисунок 6.2 — Діаграма прецедентів системи

6.2 Концептуальна модель бази даних

База даних системи складається з чотирьох таблиць реляційної бази даних, які створюють єдиний інформаційний простір для зберігання та отримання доступу до даних.

Таблиця “Користувач”, яка містить у собі інформацію про користувача системи, основну інформацію про нього та індивідуальні налаштування.

Таблиця “Кут ” містить інформацію найкращий кут в кожний місяць року.

Таблиця “Регіон” є головною, адже містить інформацію про сонячну активність в кожному регіоні.

Таблиця “Маркер” зберігає в собі координати, додаткові фото, координати та опис вказаних точок.

Концептуальна модель бази даних приведена на рисунку 4.3.

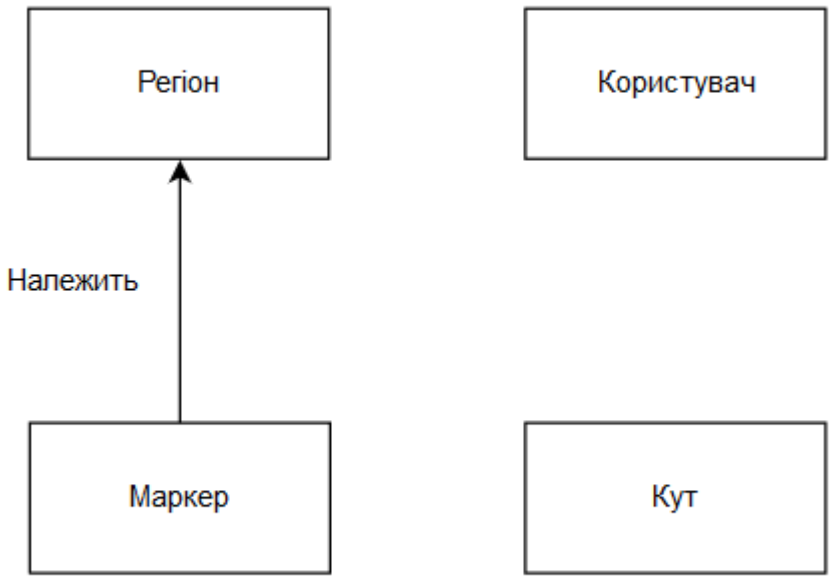


Рисунок 6.3 — Концептуальна модель БД

6.3 Опис таблиць бази даних

Для доступу до бази даних для кожної таблиці створюється клас сутності на мові C# з публічними властивостям, який є об’єктним відображенням таблиці в базі даних.

Клас DbContext забезпечує інтерфейс для доступу і CRUD операцій над даними, що зберігаються в базі даних. Клас DbSet відповідає таблиці в базі даних, поле моделі являє собою значення кожного окремого стовпця рядка.

Крім того, використання об'єктно-реляційної проекції дозволяє зручно отримувати данні з таблиць, які мають зв'язок “багато-до-багатьох” та “один-до-багатьох”.

База даних системи реалізована за допомогою ORM Entity Framework. Розглянемо більш детально структури кожної із таблиць бази даних системи розпізнавання пози людини на відеопотоці.

Детальна інформація про їх структури (ім’я, тип і розмір поля, опис поля) приведена у таблицях 4.1 — 4.4.

Таблиця 4.1. Структура таблиці “Користувач”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
UserName	nvarchar(100)	Логін користувача
Email	nvarchar(100)	Адреса електронної пошти
PasswordHash	nvarchar(10000)	Хеш паролю

Таблиця 4.2. Структура таблиці “Маркер”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
Lat	decimal	Координата широти точки
Lng	deciaml	Координата довготи точки
Text	nvarchar(450)	Текс з описом точки

Таблиця 4.3. Структура таблиці “Регіон”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
Iso	Nvarchar(10)	Исо-код регіону
Name	nvarchar(100)	Назва регіону
AvgValS	decimal	Середнє значення об’єму розсіяних променів сонця в регіоні
AvgValD	nvarchar(450)	Середнє значення об’єму прямих променів сонця в регіоні

Таблиця 4.4. Структура таблиці “Кут”

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	nvarchar(450)	Первинний ключ
Lat	Int	Широта
Grade	Int	Кут

6.4 Розробка кабінету користувача

Кабінет користувача – це основне робоче місце користувача в системі. Він включає в себе інші підмодулі, які виконують основні функції системи, та є елементом компонування в системі.

Підмодулі кабінету користувача:

До цих модулів належать наступні:

— модуль редагування даних по областям;

— модуль редагування кутового коефіцієнту;

— модуль керування картою.

6.4.1 Модуль редагування даних по областям

Даний модуль дозволяє користувачу додатку керувати даними сонячної активності з усіх регіонів:

— створювати нові та видаляти старі дані;

— редагувати вже створені;

— переглядати дані в виді таблиці.

6.4.2 Модуль редагування кутового коефіцієнту

Даний модуль дозволяє користувачу додатку керувати кутовим коефіцієнтом:

— створювати нові та видаляти старі дані;

— редагувати вже створені;

— переглядати дані в виді таблиці.

6.4.3 Модуль керування картою

Даний модуль дає користувачу доступ до карти України та встановлених маркерів:

— створювати нові та видаляти старі маркери;

— редагувати вже створені;

— переглядати карту з кольоровим відображенням сонячної активності в регіонах;

— проводити розрахунок прибутку від використання сонячних електростанцій або геліосистем в будь-якій точці України.

7. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

Розроблена програмний комплекс розроблений з використанням веб-технологій і тому працює в браузерах, які підтримують актуальні веб-стандарти.

7.1 Інсталяція та системні вимоги

Оскільки застосунок працює з використанням веб-технологій, він потребує від користувача встановити браузер та мати підключення до інтернету. Для використання необхідний веб-браузер, який підтримує актуальні веб-стандарти.

7.2 Інструкція з використання програмного продукту

При вході на клієнтський додаток, користувачеві необхідно авторизуватися в системі щоб почати працювати в системі. На рисунку 5.1 зображена форма авторизації.

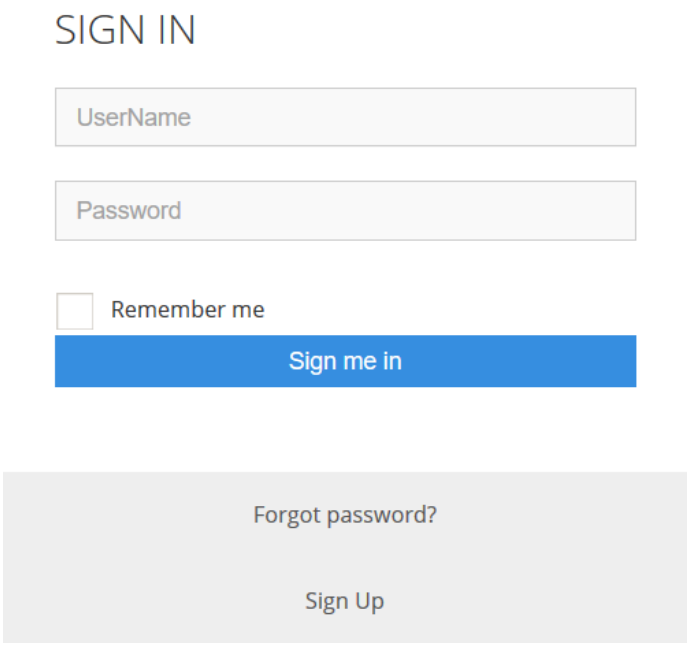


Рисунок 5.1 — Форма авторизації

У випадку якщо користувач ще ніколи не користувався системою або хоче створити новий профіль, то йому необхідно зареєструватися в системі. На рисунку 5.2 зображена форма реєстрації.

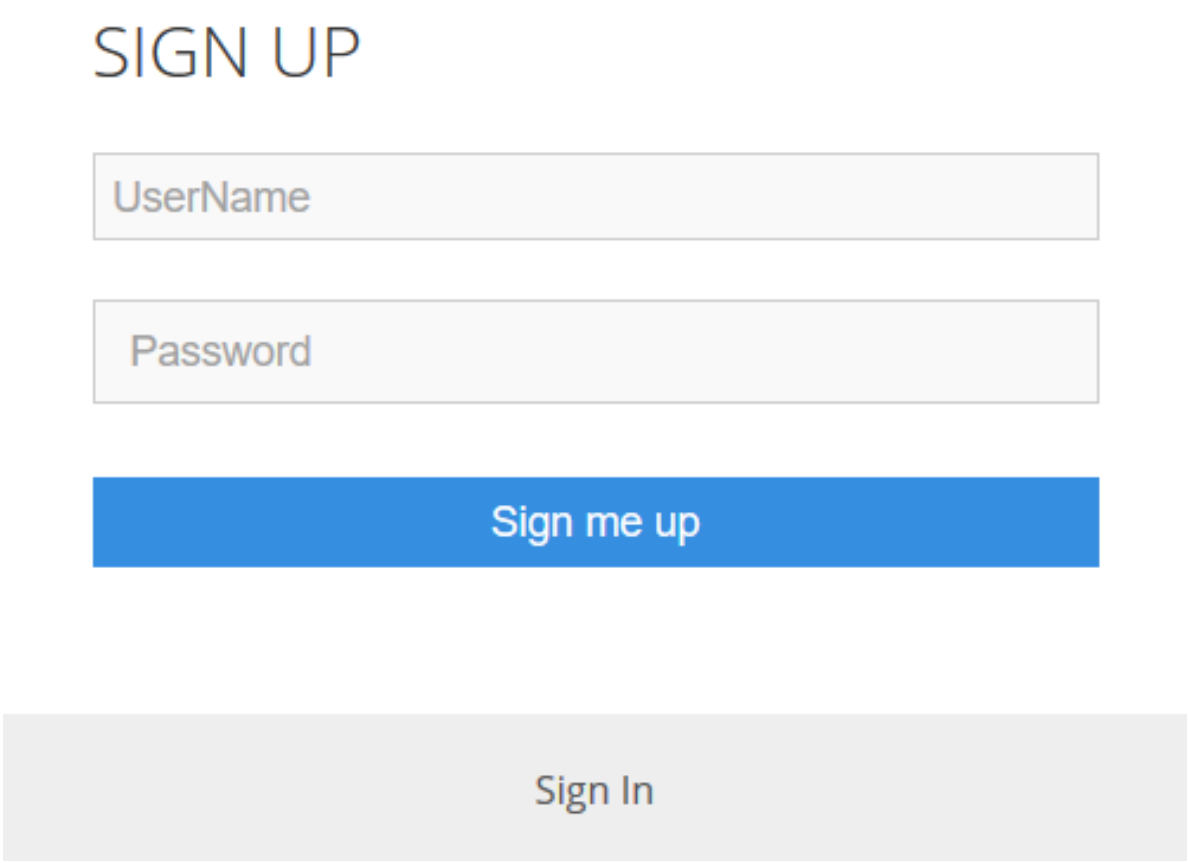


Рисунок 5.2 — Форма реєстрації

Після того, як користувач авторизувався в системі, він отримує доступ до головного меню системи. За допомогою цього меню користувач має доступ до усіх сторінок системи, або має можливість вийти з свого профілю. Також він може перейти до свого основного робочого простору – кабінету користувача. На рисунку 5.3 зображене головне меню системи.

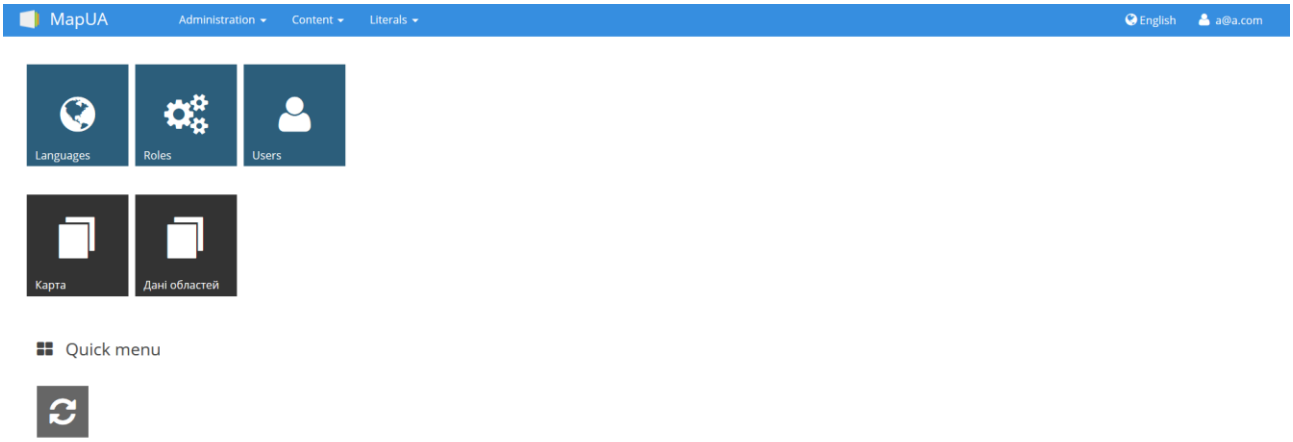
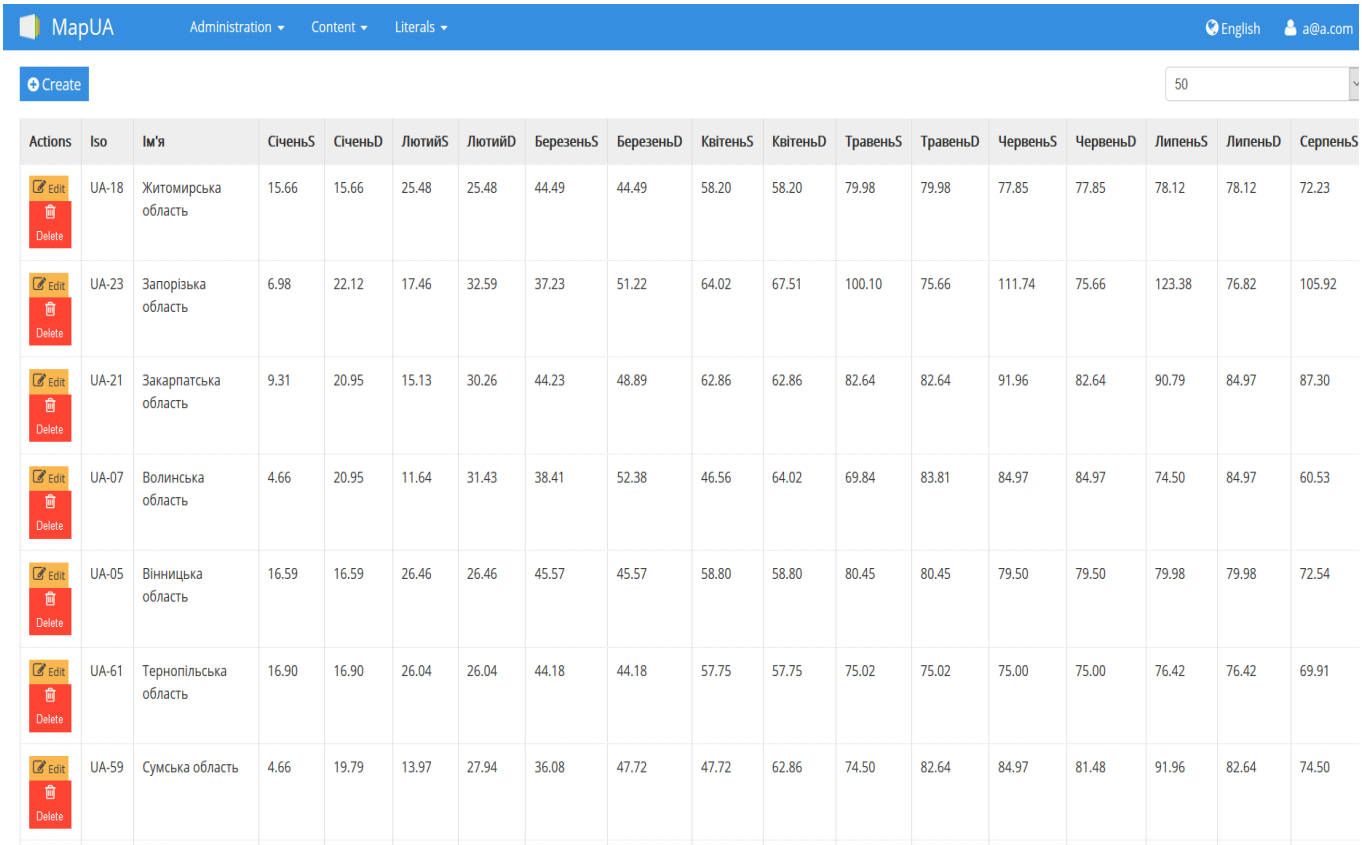


Рисунок 5.3 — Головне меню системи

Одним із компонентів кабінету користувача є меню керування даними сонячної активності в регіонаї. За його допомогою користувач має змогу створювати, редагувати та видаляти показники сонячної радіації по регіонам. На рисунку 5.4 зображене меню керування показниками сонячної радіації.



Actions	Iso	Ім'я	СіченьS	СіченьD	ЛютийS	ЛютийD	БерезеньS	БерезеньD	КвітеньS	КвітеньD	ТравеньS	ТравеньD	ЧервеньS	ЧервеньD	ЛипеньS	ЛипеньD	СерпеньS
Edit Delete	UA-18	Житомирська область	15.66	15.66	25.48	25.48	44.49	44.49	58.20	58.20	79.98	79.98	77.85	77.85	78.12	78.12	72.23
Edit Delete	UA-23	Запорізька область	6.98	22.12	17.46	32.59	37.23	51.22	64.02	67.51	100.10	75.66	111.74	75.66	123.38	76.82	105.92
Edit Delete	UA-21	Закарпатська область	9.31	20.95	15.13	30.26	44.23	48.89	62.86	62.86	82.64	82.64	91.96	82.64	90.79	84.97	87.30
Edit Delete	UA-07	Волинська область	4.66	20.95	11.64	31.43	38.41	52.38	46.56	64.02	69.84	83.81	84.97	84.97	74.50	84.97	60.53
Edit Delete	UA-05	Вінницька область	16.59	16.59	26.46	26.46	45.57	45.57	58.80	58.80	80.45	80.45	79.50	79.50	79.98	79.98	72.54
Edit Delete	UA-61	Тернопільська область	16.90	16.90	26.04	26.04	44.18	44.18	57.75	57.75	75.02	75.02	75.00	75.00	76.42	76.42	69.91
Edit Delete	UA-59	Сумська область	4.66	19.79	13.97	27.94	36.08	47.72	47.72	62.86	74.50	82.64	84.97	81.48	91.96	82.64	74.50

Рисунок 5.4 — Меню керування показниками сонячної радіації.

Головним компонентом кабінету користувача є меню керування картою. Зліва відображена інтерактивна карта, на якій можна ставити маркери, редагувати та видаляти їх.

На рисунку 5.5 зображене головне вікно керування картою.

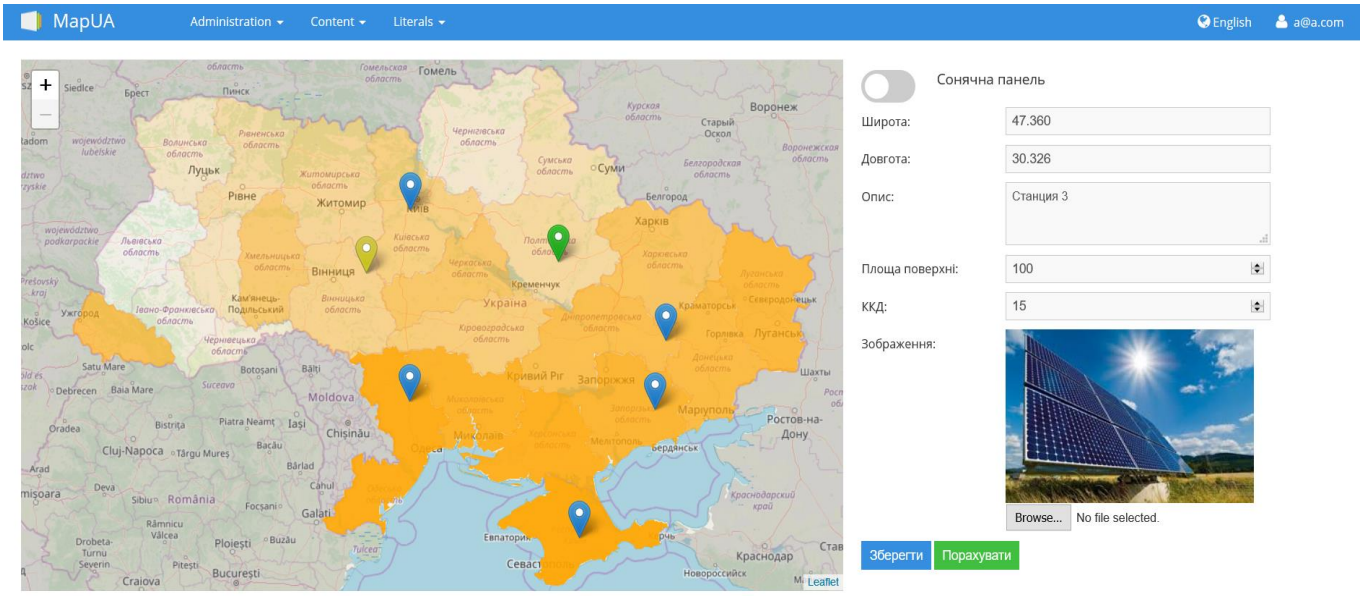


Рисунок 5.5 — Головне вікно керування картою

ВИСНОВКИ

Під час аналізу існуючого програмного забезпечення для аналізу сонячної активності було досліджено програмні застосунки, які призначені для вирішення поставленої задачі. Аналіз показав, що існуючі системи э незручними у використанні або вирішують задачу не у повному обсязі.

Було розроблено веб-додаток, що дозволить користувачу взаємодіяти з картою онлайн, ставити маркери, додавати інформацію. Крім того, користувач має змогу користуватись онлайн калькуляторами розрахунку ефективності використання сонячних панелей та геліосистем, експериментувати з показниками в різних регіонах.

Було проведено аналіз методів і засобів розробки програмних систем. Обґрунтовано вибір створення програмної системи, заснованої на веб-технологіях, а також вибір триланкової архітектури, яка дала змогу підвищити гнучкість та зручність системи, як у розробці, так і у використані.

За результатами виконання тестових завдань була підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

Користувачами системи можуть бути адміністратори веб-додатку, які відповідають за актуальність сонячних показників та наявність додаткової інформації по точкам на карті. Програмне забезпечення може бути використано на будь-якій операційній системі, де буде встановлено браузер, що підтримує сучасні веб-стандарти та має доступ до інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адам Фримен — ASP.NET Core MVC с примерами на C# для профессионалов [Електронний ресурс]. — 2017. — Режим доступу: https://bit.ly/2JbSgHe.

2. Джеймс Чамберс — ASP.NET Core. Разработка приложений [Електронний ресурс]. — 2018. — Режим доступу: https://bit.ly/2AliYL2.

3. Markus Egger — MVC Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. — 2012. — Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprise-architectures-silverlight-and-wpf.

4. Чаффер Д. Изучаем Web 1.3. Эффективная веб-разработка на JavaScript; Символ-плюс [Електронний ресурс]. — 2015. — Режим доступу: https://amzn.to/2R2rjZx.

5. Martin Fowler — Asp.Net Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: https://bit.ly/2CvCk1e.

6. Рєзцов В.Ф., Матях С.В., Кудреватих О.О. Інтерактивна карта потенціалу сонячної енергії України /Відновлювана енергетика. – 2018. – № 4 (55). – С. 34-42.

7. Martin Fowler — Asp.Net Architectures. Часть 2 [Електронний ресурс]. — 2009 — Режим доступу: https://habr.com/post/53536/.

8. Болье А. — Learning SQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.

ДОДАТОК А

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19 Б 12-1	MapRepo.cs AppDbContext.cs MapController.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_18Б 13-1	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19Б 12-1

Аркушів 7

Київ 2019

```
using Bs.Cms.Core.EntityFramework;
using MapUA.Core.EntityFramework.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Text;

namespace MapUA.Core.EntityFramework
{
    public class AppDbContext : CmsDbContextBase
    {
        private readonly AppDbContextModelProvider _modelProvider;

        public AppDbContext(DbContextOptions<AppDbContext> options,
AppDbContextModelProvider modelProvider) :
            base(options)
        {
            _modelProvider = modelProvider;
        }

        public DbSet<Mark> Marks { get; set; }

        public DbSet<Region> Regions { get; set; }

        public DbSet<Angle> Angles { get; set; }

        public DbSet<SystemStation> SystemStations { get; set; }
```

```
        public DbSet<SolarStation> SolarStations { get; set; }

        public DbSet<HelioStation> HelioStations { get; set; }
    }
}
using MapUA.Core.EntityFramework.Models;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace MapUA.Core.Repo
{
    public interface IMapRepo
    {
        Task<List<Mark>> GetAllMarks();

        Task<Mark> AddMark(Mark mark);

        Task<Mark> AddOrUpdateMark(Mark mark, IFormFile picture = null);

        Task<List<Region>> GetAllRegions();

        Task<List<Angle>> GetAllAngles();

        Task<List<SolarStation>> GetSolarStations();

        Task<List<HelioStation>> GetHelioStations();
```

```
    }
}
using MapUA.Core.EntityFramework;
using MapUA.Core.EntityFramework.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Linq;

namespace MapUA.Core.Repo
{
    public class MapRepo : IMapRepo
    {
        private readonly AppDbContext _context;

        public MapRepo(AppDbContext context)
        {
            _context = context;
        }

        public async Task<Mark> AddMark(Mark mark)
        {
            if ((await _context.Marks.FirstOrDefaultAsync(x => x.Lat == mark.Lat && x.Lng
== mark.Lng)) == null)
```

```
    {
        await _context.Marks.AddAsync(mark);
        await _context.SaveChangesAsync();
        return mark;
    }
    return null;
}

public async Task<Mark> AddOrUpdateMark(Mark mark, IFormFile picture = null)
{
    byte[] image = null;
    if(picture != null && picture.Length > 0)
    {
        using (var ms = new MemoryStream())
        {
            await picture.CopyToAsync(ms);
            image = ms.ToArray();
        }
    }
    var entity = await _context.Marks
        .AsNoTracking()
        .FirstOrDefaultAsync(x => x.Id == mark.Id);
    if(entity != null)
    {
        if(image != null)
        {
            mark.Image = image;
        }
        else
```

```
                {
                    mark.Image = entity.Image;
                }
                var entry = _context.Marks.Update(mark);
                await _context.SaveChangesAsync();
                return entry.Entity;
            }
            else if (entity == null && (await _context.Marks.FirstOrDefaultAsync(x =>
x.Lat == mark.Lat && x.Lng == mark.Lng)) == null)
            {
                mark.Image = image;
                await _context.Marks.AddAsync(mark);
                await _context.SaveChangesAsync();
                return mark;
            }
            return null;
        }

        public async Task<List<Angle>> GetAllAngles()
        {
            return await _context.Angles
                .AsNoTracking()
                .ToListAsync();
        }

        public async Task<List<Mark>> GetAllMarks()
        {
            return await _context.Marks
                .AsNoTracking()
```

```
                .Include(x => x.SystemStation)
                .ToListAsync();
        }

        public async Task<List<Region>> GetAllRegions()
        {
            return await _context.Regions
                .AsNoTracking()
                .OrderBy(x => x.AvgValS + x.AvgValD)
                .ToListAsync();
        }

        public async Task<List<HelioStation>> GetHelioStations()
        {
            return await _context.HelioStations
                .AsNoTracking()
                .ToListAsync();
        }

        public async Task<List<SolarStation>> GetSolarStations()
        {
            return await _context.SolarStations
                .AsNoTracking()
                .ToListAsync();
        }
    }
}
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Bs.Cms.UI.Security;
using MapUA.Core.EntityFramework.Models;
using MapUA.Core.Repo;
using MapUA.Core.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace MapUA.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(SecurityDefaults.DefaultPolicy)]
    public class MapController : Controller
    {
        private readonly IMapRepo _mapRepo;
        private readonly IMapper _mapper;

        public MapController(IMapRepo mapRepo, IMapper mapper)
        {
            _mapRepo = mapRepo;
            _mapper = mapper;
        }

        [HttpPost]
        public async Task<ActionResult> Mark(MarkVM model)
        {
            var picture = Request.Form.Files["markPictureName"];
```

```
            if(model.SystemStationId == null)
            {
                model.SystemStationId = 0;
            }
            if (await _mapRepo.AddOrUpdateMark(_mapper.Map<Mark>(model), picture) ==
null)
            {
                ModelState.AddModelError("", "Mark with these coordinates already
exist");
                return BadRequest(ModelState);
            }
            return RedirectToAction("Index");
        }

        public async Task<IActionResult> Index()
        {
            ViewData["Regions"] = _mapper.Map<IEnumerable<RegionVM>>(await
_mapRepo.GetAllRegions());
            ViewData["Angles"] = _mapper.Map<IEnumerable<AngleVM>>(await
_mapRepo.GetAllAngles());
            ViewData["SolarStations"] = _mapper.Map<IEnumerable<SolarStationVM>>(await
_mapRepo.GetSolarStations());
            ViewData["HelioStations"] = _mapper.Map<IEnumerable<HelioStationVM>>(await
_mapRepo.GetHelioStations());

            return View(_mapper.Map<IEnumerable<MarkVM>>(await
_mapRepo.GetAllMarks()));
        }
    }
}
```

ДОДАТОК В

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19Б 13-1

Аркушів 8

Київ 2019

АНОТАЦІЯ

Розділ містить опис частини, яка слугує для роботи з БД, що є структурною одиницею програмного продукту, та забезпечує поєднання можливостей усіх інших модулів для виконання поставлених перед системою завдань. Призначенням БД є зберігання інформації. Модуль надає можливість отримувати необхідні дані, оновлювати та видаляти необхідні записи. Модуль написано мовою програмування C#, з використанням EntityFramework.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається один з програмних модулів системи — модуль для роботи з БД з кодом УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51146_19Б 12-1, що міститься у файлі MapRepo.cs. Модуль реалізовано за допомогою EntityFramework. Модуль призначений для управління системою, яка відповідають за керування метеоданими, які зберігаються в БД. Користувач має можливість створювати новий запис, редагувати або видаляти необхідний.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням модулю для роботи з БД є збереження інформації для клієнтської частини та безпосереднього прийняття даних введення та обмін інформацією із клієнтським інтерфейсом. Використання такого шаблону дозволяє створювати програмне забезпечення, де інтерфейс і логіка роботи модуля для БД є незалежними компонентами, що дає можливість використовувати його для зменшення навантаження на клієнтську частину системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Слідкувати за зміною інформації про підсистеми є головним завданням модуля. При запуску системи модуль повертає всю інформацію, яка міститься в БД, для відображення даних на користувацькому інтерфейсі. Також модуль оброблює інформацію, надаючи змогу додавати запис, змінювати або видаляти необхідний.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблено у середовищі розробки Microsoft Visual Studio 2017, що забезпечує набір сервісних функцій та графічний діалог з користувачем, на комп’ютері, має встановлений браузер та підключення до інтернету. З БД, комп’ютер з’єднувався за допомогою пакету SqlClient.

В якості СКБД було використано PostgreSQL Server.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль реалізований як окремий клас, який забезпечує існування клієнтської частини та бізнес-логіки окремо від одного, але разом із цим запуск обох компонентів відбувається одночасно.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є інформація, яку користувач вводить в додатку.

Вихідними даними програмного модуля є необхідні записи, які потрібні для користувача.

ДОДАТОК Г

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Довідки про впровадження результатів роботи

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51166_19Б 13-1

Аркушів 1

Київ 2019